Introduction to Big Data Assignment-2

Name: Faizan Mulla
Roll No: 21F1003885

Problem Statement

The goal of this project was to develop an automated solution for counting the lines of text files uploaded to a Google Cloud Storage (GCS) bucket. By utilizing Google Cloud Functions, the solution leverages serverless architecture to handle file uploads efficiently and scalably. This report outlines the steps taken to set up the project, enable necessary APIs, write and deploy the Python code, and test the functionality.

Solution

The solution involves setting up a Google Cloud Function that triggers when a new text file is uploaded to a designated GCS bucket.
The function reads the file, counts the number of lines, and stores the result in a new file within the same bucket. This approach automates the line-counting process, making it suitable for large-scale applications where multiple files need processing without manual intervention.

Implementation Details

1. GCP Environment Setup

- **Create GCP Project:** Accessed Google Cloud Console and created a **new** project named "**IBD-GA3**". Enabled necessary APIs including *Cloud Functions API*, *Cloud Build API*, and *Cloud Storage API*.

- **Cloud Storage Setup**: Created a GCS bucket named **"ibd-ga3-bucket"** in the "**asia-south1**" region, configured with the standard storage class.

2. Cloud Function Configuration

- Create Cloud Function: Navigated to Cloud Functions in the GCP Console, created a new function with the following details:
    - **Function Name**: "count-lines"
    - **Region**: asia-south1
    - **Trigger Type**: Cloud Storage
    - **Event Type**: Finalize/Create
    - **Bucket**: Selected "ibd-ga3-bucket"
    - Set the runtime environment to **Python 3.10** and define the entry point as `**count_lines**`.

3. Write Function Code

- Developed the following Python code to count the lines in a text file:

```python
from google.cloud import storage


def count_lines(event, context):
    """Triggered by a change to a Cloud Storage bucket."""
    file_name = event["name"]
    bucket_name = event["bucket"]

    # Ignore files that start with "results_"
    if file_name.startswith("results_"):
        print(f"Ignoring file {file_name} as it starts with 'results_'")
        return

    # Initialize client
    storage_client = storage.Client()

    # Get bucket and file
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.get_blob(file_name)

    # Read content and count lines
    content = blob.download_as_text()
    line_count = len(content.splitlines())

    # Create result message
    result = f"File {file_name} contains {line_count} lines"
    print(result)  # This will appear in logs

    # Save result to new file
    result_blob = bucket.blob(f"results_{file_name}")
    result_blob.upload_from_string(result)
```

4. Create `requirements.txt`

- Added the required library to interact with GCS: “*google-cloud-storage>=2.0.0*”

Execution Process

1. Deploy Cloud Function

- Clicked "Deploy" in the Cloud Functions console.
- Waited for the function to deploy successfully, which took a few minutes.

2. Test the Function

- Uploaded a text file (`input.txt`) to the "ibd-ga3-bucket".
- Waited for the function to process the file automatically.
- Observed the creation of a new file `results_input.txt` in the bucket containing the line count result.
- Verified the output by downloading `results_input.txt`, which displayed the correct line count of the uploaded file.

Results

The function successfully counts the lines of any uploaded text file in the specified bucket and stores the results in a new file **prefixed with "results_"**.

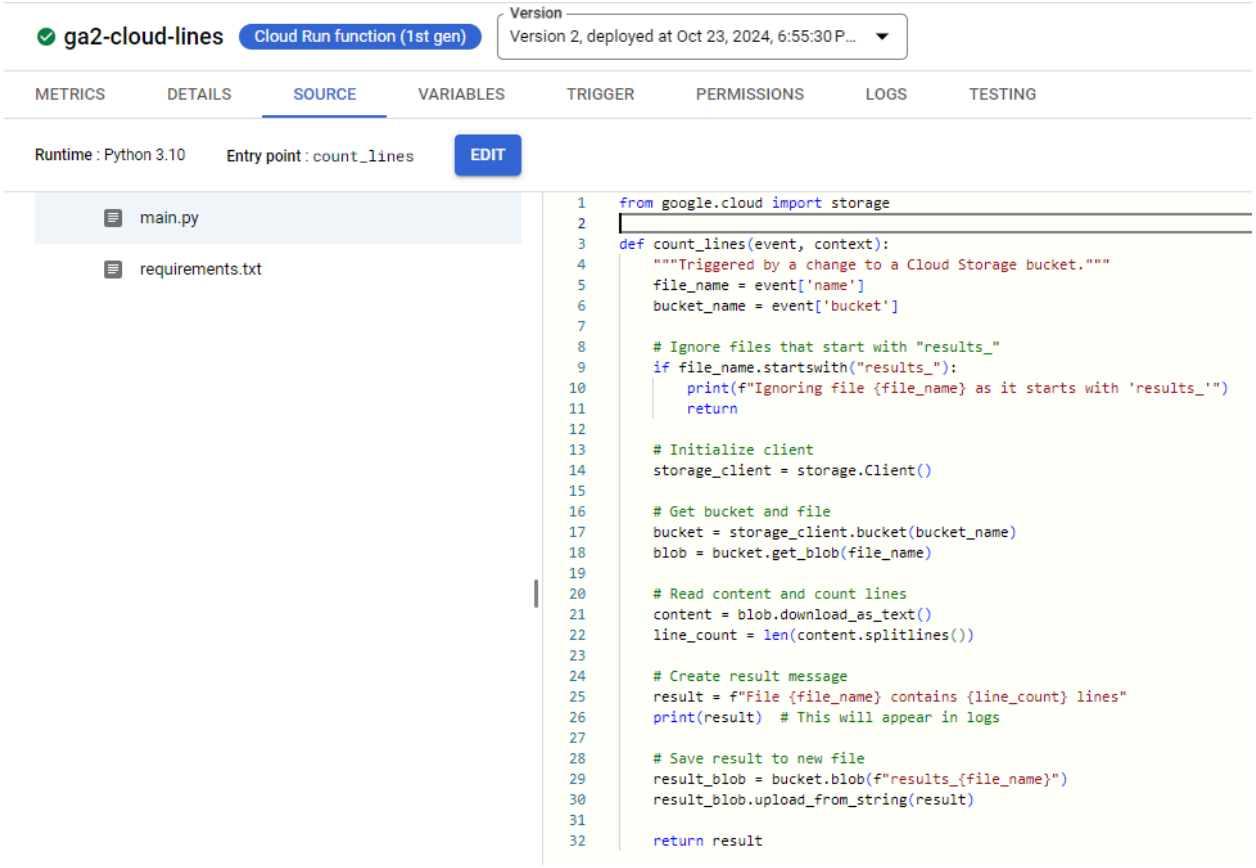
We can now just download it and add it to the submission folder.

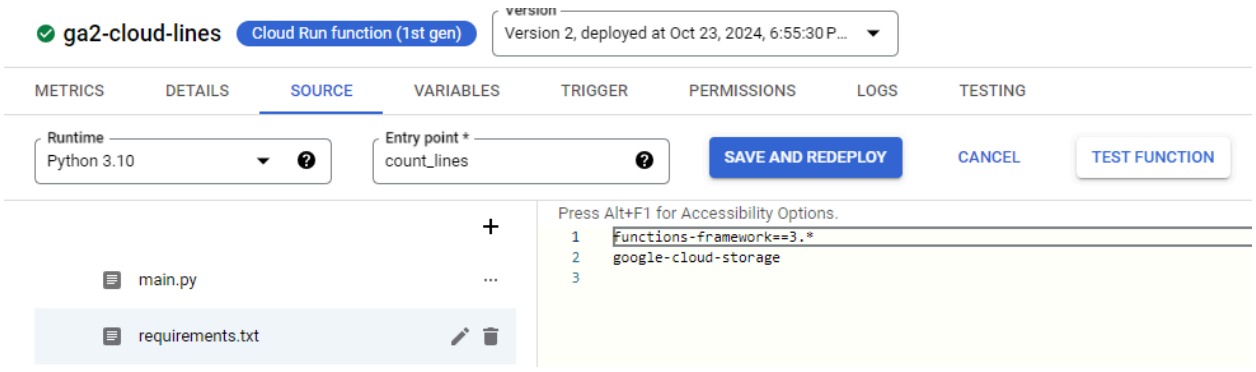
Example Result:
For a file named `input.txt` with 9 lines, the output stored in `results_input.txt` would be:

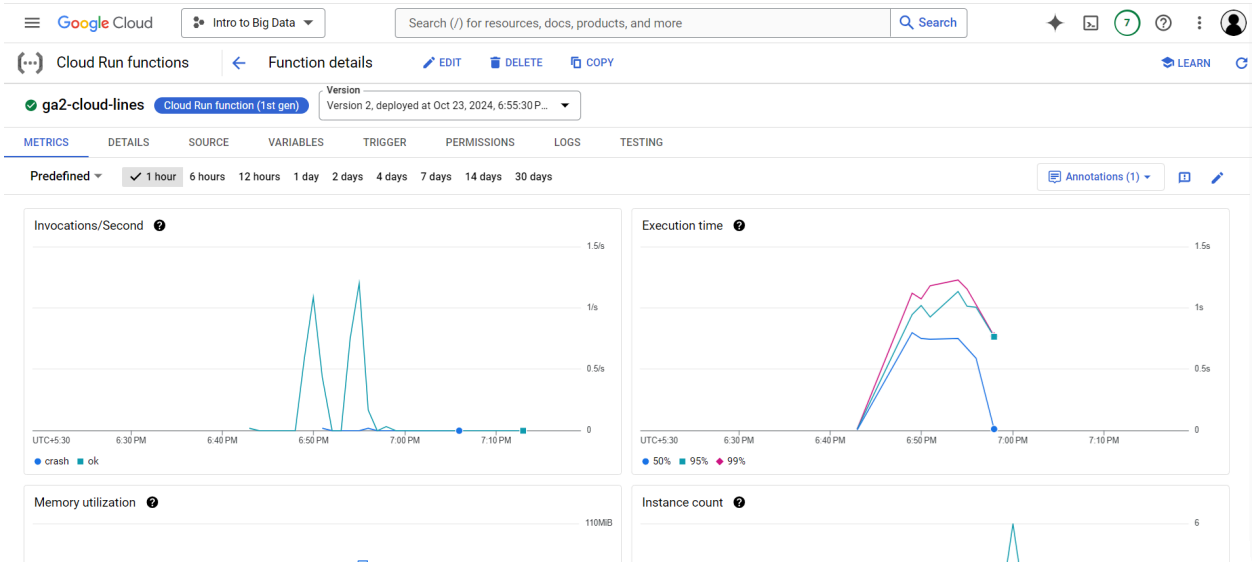
```
File example.txt contains 9 lines
```

Relevant Screenshots

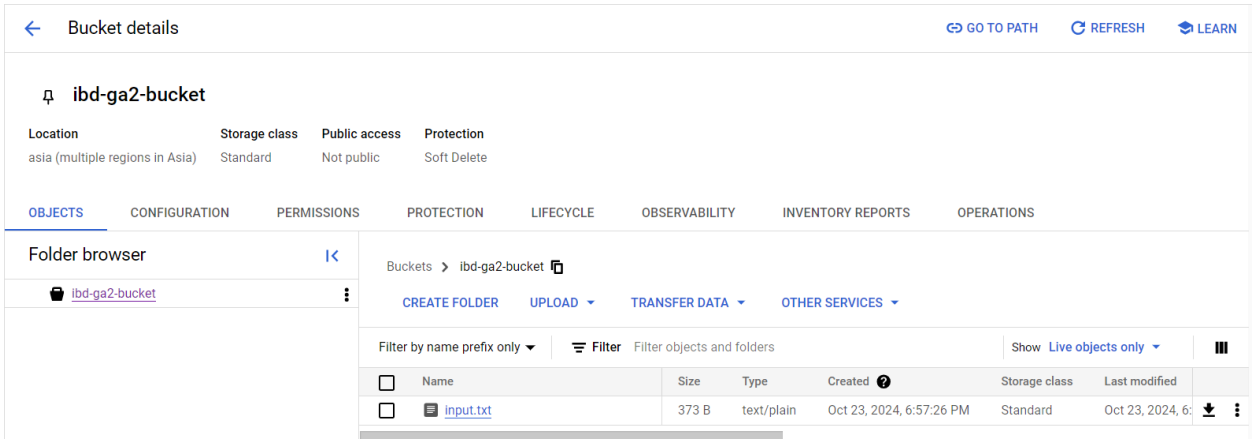
1. Cloud Function Configuration:



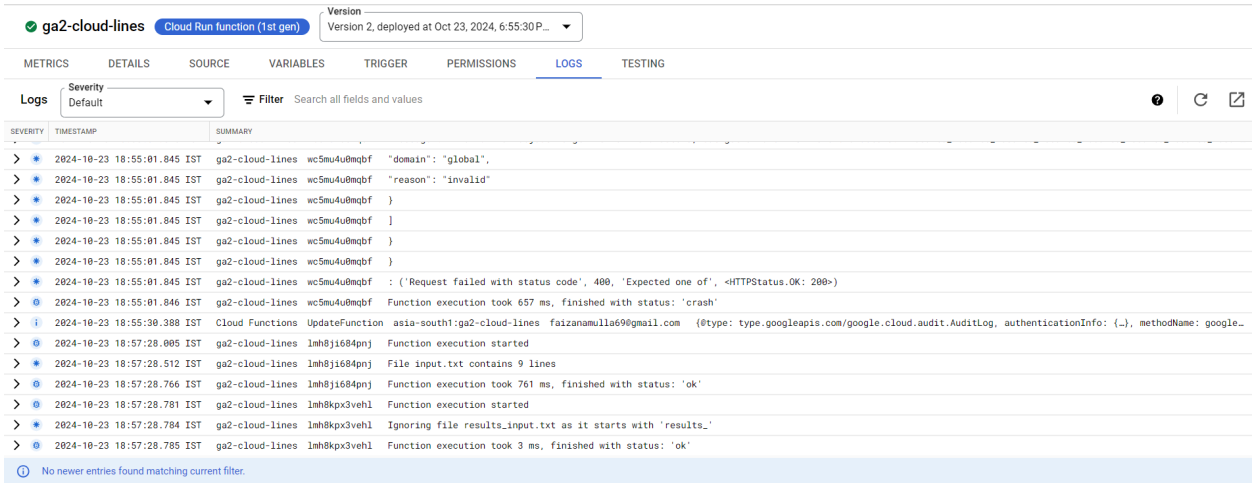




2. Storage Bucket Setup:



3. Function Execution Logs:



4. Generated Output File: